Other Operations

Output:

```
[1 2 3 4]
[1 2 3 4]
[-1. 0. 1. 2.]
[ 2. 4. 6. 8.]
[ 2 3 6 13 28]
```

1. It gives you 10000 loops. The slowest run time is 9.12 times longer than the fastest.

Output:

```
array([[ 1., 1., 1.],
[ 1., 1., 1.],
[ 1., 1., 1.]])
```

Output:

```
[1, 2, 4, 8, 16]
```

Output:

```
array([ 2, 0, 6, 4, 10])
```

2. The transpose is a view but if you use a += a.T seems to make the matrix symmetric regardless. It might not work for other operations.

Reductions

Output:

```
[[1 2 3]
[4 5 6]]
...
[1 2 3 4 5 6]
...
[[1 4]
[2 5]
[3 6]]
...
[1 4 2 5 3 6]
...
[[1 2 3]
  [4 5 6]]
```

Output:

```
[[1 2 3]
[4 5 6]]
[1 2 3 4 5 6]
[[1 2 3]
     [4 5 6]]
```

1. cumsum provides cumulatively sum of all array elements.

Shape Manipulation

1. The ravel return a view of the input. Flatten returns a copy of the elements and copy is made when needed.

Output:

```
[[1 2 3]
[4 5 6]]
...
[[1 4]
[2 5]
[3 6]]
```

Output:

```
array([[4, 7, 9],
       [1, 2, 8]])
```

Output:

```
(array([[4, 7, 9],
[1, 2, 8]]), array([[4, 7, 9],
       [1, 2, 8]]))
```

Output:

```
array([['3.0', '4', '6.32', 'a'],
['123', '5463', '9.23563', 'ehehehe']],
       dtype='|S7')
```

Output:

```
       (array([[43, 22, 1, 2],
       [ 6, 5, 32, 21],
       [42, 57, 13, 96]]), array([[ 1, 2, 22, 43],
       [ 5, 6, 21, 32],
       [13, 42, 57, 96]]))
```

Output:

```
       (array([[ 6, 5, 32, 21],
       [43, 22, 1, 2],
       [42, 57, 13, 96]]),
       array([ 6, 5, 32, 21, 43, 22, 1, 2, 42, 57, 13, 96]),
       array([[ 5, 6, 21, 32],
       [ 1, 2, 22, 43],
       [13, 42, 57, 96]]),
```

array([6, 5, 32, 21, 43, 22, 1, 2, 42, 57, 13, 96]),
array([1, 2, 5, 6, 13, 21, 22, 32, 42, 43, 57, 96]))